

### Prueba 1 - Practica

# Objetivo:

• Consolidar los conocimientos adquiridos en clase para la programación de una aplicación en Python.

# Enunciado:

 Realizar una aplicación que permita gestionar los pedidos de una empresa almacenando la información dentro de archivos o base de datos (SQLite), para lo cual deben seguir los siguientes paso:

La aplicación deberá: manejar clientes (se guarda su nombre, dirección, teléfono y e-mail), que pueden realizar pedidos de productos, de los cuales se anota la cantidad en stock. Un cliente puede tener una o varias cuentas para el pago de los pedidos. Cada cuenta está asociada a una tarjeta de crédito, y tiene una cierta cantidad disponible de dinero, que el cliente debe aumentar periódicamente para poder realizar nuevos pedidos.

Un cliente puede empezar a realizar un pedido sólo si tiene alguna cuenta con dinero disponible. Además, sólo es posible realizar peticiones de productos en stock.

Existe una clase responsable del cobro, orden de distribución y confirmación de los pedidos. El cobro se realiza al finalizar el pedido. Si una cuenta no tiene suficiente dinero, el pedido se rechaza. Una vez que el pedido está listo para servirse, se ordena su distribución, y una vez entregado, pasa a estar confirmado.

\* Se aprobará como puntos adicionales a la practicas si se realiza una implementación visual utilizando cualquier librería GUI (Tkinter – 1 punto) o mejor aún con Flask(3 puntos).

Finalmente, exportar un PDF del cuaderno de Jupyter Notebook visualizando el funcionamiento y validación del sistema.

**Plazo**: Se debe presentar el sistema funcionando hasta las **23:55 del 09/05/2021**, la misma que deberá ser subida al git personal y adicionalmente dentro de un cuaderno de Jupyter Notebook.

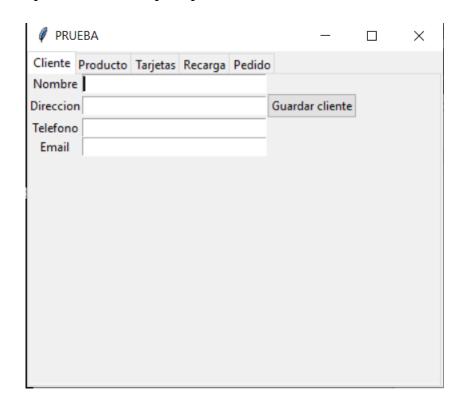


## Prueba 1 - Practica

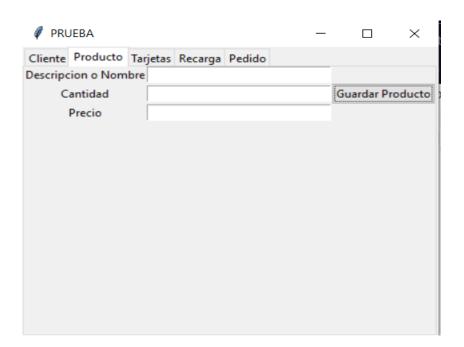
Nombre: Carlos Alvarez

URL GITHUB: https://github.com/dani-alv97/Prueba1-IA

• Interfaz Grafica primera ventana para poder crear los Clientes



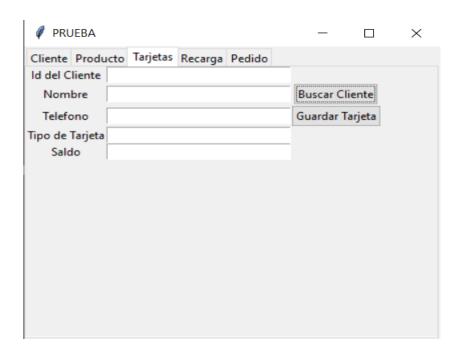
• Interfaz Grafica segunda ventana para poder crear Productos



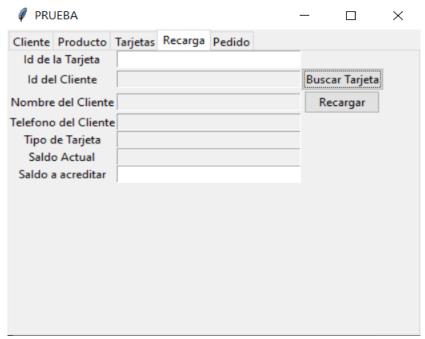


## Prueba 1 - Practica

• Interfaz Grafica tercera ventana para poder crear Tarjetas



• Interfaz Grafica cuarta ventana para poder recargar el saldo de las tarjetas

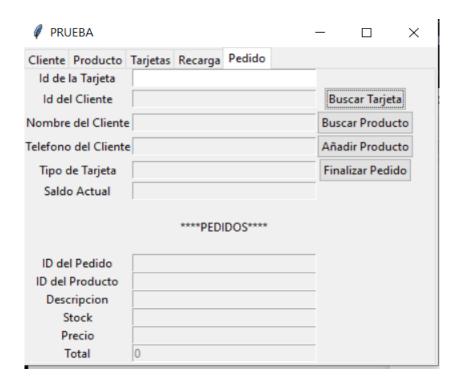


• Interfaz Grafica quinta ventana para poder crear un pedido buscando el cliente y el producto

**Tema**: Inteligencia Artificial 1.



### Prueba 1 - Practica



# • Código de las funciones

```
In [1]: import tkinter as tk
         from tkinter import ttk
from tkinter import messagebox
         import sqlite3
In [2]: conexion=sqlite3.connect("pruebaia.db")
         cursor = conexion.cursor()
In [3]: tabla1 = """
                                    create table if not exists clientes (cliente integer primary key autoincrement,
                                                               nombre text, direccion text, telefono integer, email text)
         cursor.execute(tabla1)
         tabla2 = """
                                    create table if not exists productos (producto integer primary key autoincrement, descripcion text, cantidad integer, precio double, pedido integer, foreign key(pe
         cursor.execute(tabla2)
tabla3 = """
                                    create table if not exists tarjetas (id_tarjeta integer primary key autoincrement, cliente integer,
                                                                            tipo text, saldo double, foreign key(cliente) references clientes(cli
         cursor.execute(tabla3)
         tabla4 = '
                                    create table if not exists pedidos (id_pedido integer primary key autoincrement, cliente integer, total of
                                                                            foreign key(cliente) references clientes(cliente))
         cursor.execute(tabla4)
```

# Simulación Tema: Inteligencia Artificial 1.



```
In [4]: class Metodos:
               def crearCliente(self):
                    confirmacion = messagebox.askyesno(message="¿Desea crear el cliente?", title="Cliente")
                    if confirmacion == True:
                         nombre = self.nombreCliente.get()
                          direccion=self.direccionCliente.get()
                         telefono=self.telefonoCliente.get()
                         email=self.emailCliente.get()
                         cursor.execute('INSERT INTO clientes (nombre, direccion, telefono, email) VALUES(?, ?, ?, ?)', (nombre, direccion, telefono, email)
                         conexion.commit()
                         self.nombreCliente.delete(0, 'end')
                         self.direccionCliente.delete(0, 'end')
self.telefonoCliente.delete(0, 'end')
                         self.emailCliente.delete(0, 'end')
messagebox.showinfo(message="Cliente Creado", title="Cliente")
                    if confirmacion == False:
                         self.nombreCliente.delete(0, 'end')
                         self.direccionCliente.delete(0, 'end')
self.telefonoCliente.delete(0, 'end')
                         self.emailCliente.delete(0, 'end')
messagebox.showinfo(message="No se a creado al cliente", title="Cliente")
                    if confirmacion == None:
                         self.nombreCliente.delete(0, 'end')
self.direccionCliente.delete(0, 'end')
self.telefonoCliente.delete(0, 'end')
                         self.emailCliente.delete(0, 'end')
messagebox.showinfo(message="Vuelva a ingresar los datos del cliente", title="Cliente")
               def crearProducto(self):
                     confirmacion = messagebox.askyesno(message="¿Desea crear el producto?", title="Producto")
                    if confirmacion == True:
    descripcion = self.descripcionProducto.get()
                         cantidad = self.cantidadProducto.get()
                         precio = self.precioProducto.get()
                         cursor execute('INSERT INTO productos (descripcion, cantidad, precio) VALUES(?, ?, ?)', (descripcion, cantidad, preci
                         conexion.commit()
                         self.descripcionProducto.delete(0, 'end')
                         self.cantidadProducto.delete(0, 'end')
self.precioProducto.delete(0, 'end')
messagebox.showinfo(message="Producto Creado", title="Producto")
                    if confirmacion == False:
                         self.descripcionProducto.delete(0, 'end')
                         self.cantidadProducto.delete(0, 'end')
self.precioProducto.delete(0, 'end')
                         messagebox.showinfo(message="No se a creado el producto", title="Producto")
                    if confirmacion == None:
                         self.descripcionProducto.delete(0, 'end')
                         self.cantidadProducto.delete(0, 'end')
self.precioProducto.delete(0, 'end')
                         self.precioProducto.delete(0, 'end')
messagebox.showinfo(message="Vuelva a ingresar los datos del cliente", title="Producto")
```

Tema: Inteligencia Artificial 1.



```
def buscarPedido(self):
    self.idPedido.config(state=tk.NORMAL)
    self.idPedido.delete(0, 'end')
    idPedido = self.idPedido.get()
    cursor.execute("SELECT id_pedido FROM pedidos ORDER BY id_pedido DESC LIMIT 1;")
    fila = cursor.fetchone()
    self.idPedido.insert(0, fila[0])
    self.idPedido.config(state=tk.DISABLED)
def buscarTarjeta(self):
    self.tipoTarjeta3.config(state=tk.NORMAL)
    self.saldoTarjeta3.config(state=tk.NORMAL)
    self.idClienteTarjeta3.config(state=tk.NORMAL)
    self.nombreCliente3.config(state=tk.NORMAL)
    self.telefonoCliente3.config(state=tk.NORMAL)
   self.tipoTarjeta3.delete(0, 'end')
self.saldoTarjeta3.delete(0, 'end')
    self.idClienteTarjeta3.delete(0, 'end')
    self.nombreCliente3.delete(0, 'end')
    self.telefonoCliente3.delete(0,
    idTarjeta = self.idTarjeta3.get()
    tipo = self.tipoTarjeta3.get()
    saldo = self.saldoTarjeta3.get()
    idCliente = self.idClienteTarjeta3.get()
    nombre = self.nombreCliente3.get()
    telefono = self.telefonoCliente3.get()
    cursor.execute("SELECT tar.id_tarjeta, tar.tipo, tar.saldo, tar.cliente, cli.nombre, cli.telefono FROM clientes cli, tar
    fila = cursor.fetchone()
    self.tipoTarjeta3.insert(0, fila[1])
    self.saldoTarjeta3.insert(0, fila[2])
    self.idClienteTarjeta3.insert(0, fila[3])
    self.nombreCliente3.insert(0, fila[4])
    self.telefonoCliente3.insert(0, fila[5])
    self.tipoTarjeta3.config(state=tk.DISABLED)
    self.saldoTarjeta3.config(state=tk.DISABLED)
    self.idClienteTarjeta3.config(state=tk.DISABLED)
    self.nombreCliente3.config(state=tk.DISABLED)
    self.telefonoCliente3.config(state=tk.DISABLED)
def buscarTarjeta2(self):
    self.buscarPedido()
    idPedido = int(self.idPedido.get()) + 1
    self.tipoTarjeta2.config(state=tk.NORMAL)
    self.saldoTarjeta2.config(state=tk.NORMAL)
    self.idClienteTarjeta2.config(state=tk.NORMAL)
    self.nombreCliente2.config(state=tk.NORMAL)
    self.telefonoCliente2.config(state=tk.NORMAL)
    self.idPedido.config(state=tk.NORMAL)
    self.tipoTarjeta2.delete(0, 'end')
    self.idPedido.delete(0, 'end')
    self.saldoTarjeta2.delete(0, 'end')
    self.idClienteTarjeta2.delete(0, 'end')
    self.nombreCliente2.delete(0, 'end')
    self.telefonoCliente2.delete(0, 'end')
    idTarjeta = self.idTarjeta2.get()
    tipo = self.tipoTarjeta2.get()
    saldo = self.saldoTarjeta2.get()
    idCliente = self.idClienteTarjeta2.get()
    nombre = self.nombreCliente2.get()
    telefono = self.telefonoCliente2.get()
```

# 3

### Simulación

Tema: Inteligencia Artificial 1.



```
def buscarTarjeta2(self):
    self.buscarPedido()
    idPedido = int(self.idPedido.get()) + 1
    self.tipoTarjeta2.config(state=tk.NORMAL)
    self.saldoTarjeta2.config(state=tk.NORMAL)
    self.idClienteTarjeta2.config(state=tk.NORMAL)
    self.nombreCliente2.config(state=tk.NORMAL)
    self.telefonoCliente2.config(state=tk.NORMAL)
    self.idPedido.config(state=tk.NORMAL)
    self.tipoTarjeta2.delete(0, 'end')
    self.idPedido.delete(0, 'end')
    self.saldoTarjeta2.delete(0, 'end')
    self.idClienteTarjeta2.delete(0, 'end')
    self.nombreCliente2.delete(0, 'end')
    self.telefonoCliente2.delete(0, 'end')
    idTarjeta = self.idTarjeta2.get()
    tipo = self.tipoTarjeta2.get()
    saldo = self.saldoTarjeta2.get()
    idCliente = self.idClienteTarjeta2.get()
    nombre = self.nombreCliente2.get()
    telefono = self.telefonoCliente2.get()
    cursor.execute("SELECT tar.id_tarjeta, tar.tipo, tar.saldo, tar.cliente, cli.nombre, cli.telefono FROM clientes cli, tarj
    fila = cursor.fetchone()
    self.tipoTarjeta2.insert(0, fila[1])
    self.saldoTarjeta2.insert(0, fila[2])
    self.idClienteTarjeta2.insert(0, fila[3])
    self.nombreCliente2.insert(0, fila[4])
    self.telefonoCliente2.insert(0, fila[5])
    if float(self.saldoTarjeta2.get()) > 0.0:
        self.idProducto4.config(state=tk.NORMAL)
        {\tt self.descripcionProducto4.config(state=tk.NORMAL)}
        self.stockProducto4.config(state=tk.NORMAL)
        self.precioProducto4.config(state=tk.NORMAL)
        self.idPedido.config(state=tk.NORMAL)
        self.idPedido.insert(0, idPedido)
        messagebox.showinfo(message="SU SALDO ES MAYOR A 0")
        cliente4 = self.idClienteTarjeta2.get()
        cursor.execute('INSERT INTO pedidos (cliente) VALUES(?)', (cliente4))
        conexion.commit()
    else :
        messagebox.showinfo(message="SU SALDO ES 0")
    self.idPedido.config(state=tk.DISABLED)
    self.tipoTarjeta2.config(state=tk.DISABLED)
    self.saldoTarjeta2.config(state=tk.DISABLED)
    self.idClienteTarjeta2.config(state=tk.DISABLED)
    self.nombreCliente2.config(state=tk.DISABLED)
    self.telefonoCliente2.config(state=tk.DISABLED)
    self.descripcionProducto4.config(state=tk.DISABLED)
    self.stockProducto4.config(state=tk.DISABLED)
    self.precioProducto4.config(state=tk.DISABLED)
def buscarProducto(self):
    self.descripcionProducto4.config(state=tk.NORMAL)
    self.stockProducto4.config(state=tk.NORMAL)
    self.precioProducto4.config(state=tk.NORMAL)
   self.descripcionProducto4.delete(0, 'end')
    self.stockProducto4.delete(0, 'end')
   self.precioProducto4.delete(0, 'end')
    idProducto = self.idProducto4.get()
   descripcion = self.descripcionProducto4.get()
   precio = self.precioProducto4.get()
   stock = self.stockProducto4.get()
   cursor.execute("SELECT producto, descripcion, cantidad, precio FROM productos WHERE producto = ?", (idProducto))
   fila1 = cursor.fetchone()
   self.descripcionProducto4.insert(0, fila1[1])
   self.stockProducto4.insert(0, fila1[2])
   self.precioProducto4.insert(0, fila1[3])
   self.descripcionProducto4.config(state=tk.DISABLED)
   self.stockProducto4.config(state=tk.DISABLED)
   self.precioProducto4.config(state=tk.DISABLED)
```

Tema: Inteligencia Artificial 1.



```
def recarga(self):
    confirmacion = messagebox.askyesno(message="¿Desea recargar la tarjeta?", title="Tarjeta")
    if confirmacion == True:
        idTarjeta = self.idTarjeta3.get()
         saldo = self.saldoTarjeta3.get()
         saldoAcreditar = self.saldoAcreditar3.get()
        nuevoSaldo = int(float(saldo)) + int(float(saldoAcreditar))
update = """UPDATE tarjetas SET saldo = ? WHERE id_tarjeta = ?"""
         data = (nuevoSaldo, idTarjeta)
         cursor.execute(update, data)
         conexion.commit()
         messagebox.showinfo(message="Recarga Realizada", title="Recarga")
         self.tipoTarjeta3.config(state=tk.NORMAL)
         self.saldoTarjeta3.config(state=tk.NORMAL)
         self.idClienteTarjeta3.config(state=tk.NORMAL)
         self.nombreCliente3.config(state=tk.NORMAL)
         self.telefonoCliente3.config(state=tk.NORMAL)
         self.idTarjeta3.delete(0, 'end')
        self.tipoTarjeta3.delete(0, 'end')
self.saldoTarjeta3.delete(0, 'end')
         self.idClienteTarjeta3.delete(0, 'end')
        self.nombreCliente3.delete(0, 'end')
self.telefonoCliente3.delete(0, 'end')
self.saldoAcreditar3.delete(0, 'end')
    if confirmacion == False:
         messagebox.showinfo(message="No se ha hecho la recarga", title="Recarga")
         self.tipoTarjeta3.config(state=tk.NORMAL)
         self.saldoTarjeta3.config(state=tk.NORMAL)
         self.idClienteTarjeta3.config(state=tk.NORMAL)
         self.nombreCliente3.config(state=tk.NORMAL)
         self.telefonoCliente3.config(state=tk.NORMAL)
        self.tipoTarjeta3.delete(0, 'end')
self.saldoTarjeta3.delete(0, 'end')
         self.idClienteTarjeta3.delete(0, 'end')
         self.nombreCliente3.delete(0, 'end')
         self.telefonoCliente3.delete(0, 'end')
    if confirmacion == None:
         messagebox.showinfo(message="Vuelva a ingresar los datos para la recarga", title="Recarga")
         self.tipoTarjeta3.config(state=tk.NORMAL)
         self.saldoTarjeta3.config(state=tk.NORMAL)
         self.idClienteTarjeta3.config(state=tk.NORMAL)
         self.nombreCliente3.config(state=tk.NORMAL)
         self.telefonoCliente3.config(state=tk.NORMAL)
        self.tipoTarjeta3.delete(0, 'end')
self.saldoTarjeta3.delete(0, 'end')
         self.idClienteTarjeta3.delete(0, 'end')
         self.nombreCliente3.delete(0,
                                           'end')
        self.telefonoCliente3.delete(0, 'end')
```

# 3

# Simulación

**Tema**: Inteligencia Artificial 1.



```
def carrito(self):
    totalNuevo = float(self.totalFinal.get())
    self.totalFinal.config(state=tk.NORMAL)
   self.totalFinal.delete(0,'end')
   self.idPedido.config(state=tk.NORMAL)
   precioProducto = float(self.precioProducto4.get())
    idPedido = self.idPedido.get()
   idProducto = self.idProducto4.get()
   print(idPedido)
   print(idProducto)
    totalNuevo = (int(float(precioProducto))) + (int(float(totalNuevo)))
   self.totalFinal.insert(0, totalNuevo)
    update = """UPDATE productos SET pedido = ? WHERE producto = ?"""
   data = (idPedido, idProducto)
    cursor.execute(update, data)
   conexion.commit()
    self.totalFinal.config(state=tk.DISABLED)
   self.idPedido.config(state=tk.DISABLED)
def pagar(self):
   self.idPedido.config(state=tk.NORMAL)
    self.totalFinal.config(state=tk.NORMAL)
    idPedido2 = int(self.idPedido.get())
    total = self.totalFinal.get()
    saldo = self.saldoTarjeta2.get()
   idTarjeta = self.idTarjeta2.get()
update = """UPDATE pedidos SET total = ? WHERE id_pedido = ?"""
    data = (total, idPedido2)
   cursor.execute(update, data)
    saldoFinal = int(float(saldo)) - int(float(total))
    if saldo > total:
       update2 = """UPDATE tarjetas SET saldo = ? WHERE id_tarjeta = ?"""
       data2 = (saldoFinal, idTarjeta)
        cursor.execute(update2, data2)
       messagebox.showinfo(message="No tiene suficiente saldo por favor haga una recarga", title="Recarga")
```

Tema: Inteligencia Artificial 1.



```
def __init__(self):
        self.ventana1=tk.Tk()
        self.ventana1.title("PRUEBA ")
        self.cuaderno1 = ttk.Notebook(self.ventana1)
        self.pagina1 = ttk.Frame(self.cuaderno1)
       self.cuaderno1.add(self.pagina1, text="Cliente")
self.label1 = ttk.Label(self.pagina1, text="Nombre")
        self.label1.grid(column=0, row=0)
        self.label1 = ttk.Label(self.pagina1, text="Direccion")
        self.label1.grid(column=0, row=1)
        self.label1 = ttk.Label(self.pagina1, text="Telefono")
        self.label1.grid(column=0, row=2)
        self.label1 = ttk.Label(self.pagina1, text="Email")
        self.label1.grid(column=0, row=3)
        self.btnCrearCliente = ttk.Button(self.paginal, text="Guardar cliente",command=self.crearCliente)
        self.btnCrearCliente.grid(column=2, row=1)
        #caias texto
        self.nombreCliente = tk.Entry(self.pagina1, width=30)
        self.nombreCliente.grid(column=1, row=0)
        self.direccionCliente = tk.Entry(self.pagina1, width=30)
        self.direccionCliente.grid(column=1, row=1)
        self.telefonoCliente = tk.Entry(self.paginal, width=30)
        self.telefonoCliente.grid(column=1, row=2)
        self.emailCliente = tk.Entry(self.pagina1, width=30)
        self.emailCliente.grid(column=1, row=3)
#ventana 2
        self.pagina2 = ttk.Frame(self.cuaderno1)
        self.cuaderno1.add(self.pagina2, text="Producto")
       self.label1=ttk.Label(self.pagina2, text="Descripcion o Nombre")
        self.label1.grid(column=0, row=0)
        self.label1=ttk.Label(self.pagina2, text="Cantidad")
        self.label1.grid(column=0, row=1)
        self.label1=ttk.Label(self.pagina2, text="Precio")
        self.label1.grid(column=0, row=2)
        self.btnCrearProducto=ttk.Button(self.pagina2, text="Guardar Producto",command=self.crearProducto)
        self.btnCrearProducto.grid(column=2, row=1)
        #caias texto
        self.descripcionProducto=tk.Entry(self.pagina2, width=30)
        self.descripcionProducto.grid(column=1, row=0)
        self.cantidadProducto=tk.Entry(self.pagina2, width=30)
        self.cantidadProducto.grid(column=1, row=1)
        self.precioProducto=tk.Entry(self.pagina2, width=30)
        self.precioProducto.grid(column=1, row=2)
```